Presentación del Framework Web Astro

Autor: Jordi Pradas López, NIU: 1599977

***Abstract*—En este paper, presentamos el framework de desarrollo web Astro, que ofrece una experiencia de desarrollo cómoda e intuitiva para la creación de páginas estáticas, rápidas y enfocadas en el contenido. Astro se centra en el desarrollo de sitios web ricos en contenido y utiliza una arquitectura de renderizado en el servidor para lograr un rendimiento óptimo. Con su enfoque en la minimización del JavaScript enviado al cliente, Astro garantiza una carga rápida de las páginas y una experiencia de usuario fluida. Además, Astro permite la integración de componentes de otros frameworks de User Interface, como React y Vue, ofreciendo flexibilidad y extensibilidad. Presentamos la estructura del proyecto en Astro, destacando las características clave y la sintaxis JavaScript Syntax Extension que ofrece. También discutimos la generación de múltiples páginas, el uso de componentes y la integración de librerías externas. Finalmente, para consolidar todo lo aprendido y servir de inspiración a quien quiera probar Astro, hemos desarrollado un pequeño proyecto y hemos publicado su código fuente.**

***Index Terms*—Alto rendimiento, arquitectura MPA, Astro, experiencia de desarrollo, fácil de usar, flexibilidad de tecnologías web, framework de desarrollo front-end, front-end components, importación de componentes, Islas de Componentes, página web estática, renderizado en el servidor, sintaxis JSX, sitio web rico en contenido, soporte para CSS, soporte para Markdown, soporte para TS.**

I. Introducción

ESTE documento pretende presentar y mostrar el potencial del *framework* de desarrollo web todo en uno, Astro. Astro ofrece una experiencia de desarrollo muy cómoda e intuitiva. Además, como descubrirán a lo largo del *paper*, las características de Astro lo hacen ideal para el desarrollo de páginas estáticas rápidas y enfocadas en el contenido, como blogs y portafolios. Los desarrolladores de Astro quisieron crear un *framework* que fuera (1) enfocado en el contenido: diseñado para sitios web ricos en contenido; (2) de renderizado en el servidor: ya que los sitios web se ejecutan más rápido cuando procesan HTML en el lado del servidor; (3) rápido por defecto: envía el mínimo JavaScript (JS) al cliente; (4) fácil de usar: utiliza un lenguaje de componentes propio, muy cercano al HTML y con elementos de otros *frameworks* como expresiones JSX y soporte para el alcance de CSS[1] por defecto (Svelte y Vue); (5) flexible y extensible: tiene todo lo necesario para realizar proyectos completos en él, pero permite integraciones de frameworks de front-end como React, Vue, etc.

[1]El alcance de CSS se refiere a la forma en que los estilos CSS se aplican y afectan a los elementos HTML en una página web. En algunos casos, cuando se utilizan componentes reutilizables o anidados, los estilos CSS pueden afectar a elementos que no se pretende que se vean afectados.

II. Empezar con Astro

A. Instalación

StackBlitz permite usar Astro en el navegador[2] o, si lo preferimos, podemos instalarlo de forma fácil como dependencia de Node Package Manager (NPM) mediante el comando:

```
npm create astro
```

Tras lanzar dicho comando, se nos preguntará cómo queremos que se denomine el proyecto; si queremos usar plantillas ya existentes o empezar desde cero; si deseamos que las dependencias de Astro se instalen automáticamente, con NPM; si planeamos usar TypeScript (TS) y, en caso afirmativo, con qué nivel de flexibilidad: estricto, muy estricto o relajado; y, finalmente, si queremos inicializar un repositorio de Git.

Una vez creado el proyecto, podemos empezar a trabajar en él con nuestro editor de código o *Integrated Development Environment* (IDE) favorito. Para una experiencia de desarrollo más cómoda, se recomienda instalar la extensión de Astro. Esta la podemos instalar de forma fácil, por ejemplo, en Visual Studio Code (VS Code) basta con buscar Astro en el panel de Extensiones e instalar la primera de todas.

B. Características Clave

Las características más importantes del *framework* son, junto a algunas de las que ya hemos mencionado en la introducción, las siguientes:

- Islas de Componentes: es un patrón de arquitectura web promovido por Astro, que utiliza componentes de *User Interface* (UI) interactivos (islas) en páginas HTML, predominantemente estáticas. Esta técnica, llamada hidratación parcial, permite utilizar componentes de cualquier *framework* de UI, como React, Vue, etc. sin ralentizar demasiado la página, ya que Astro renderizará estos componentes como HTML y quitará o reducirá al mínimo el código JS.
- Arquitectura *Multiple Page Application* (MPA) que prioriza el servidor: a diferencia de otros *frameworks*, como Vue, Astro no utiliza la arquitectura *Single Page Application* (SPA) sino que usa una arquitectura MPA que consta de varias páginas HTML, renderizadas mayoritariamente en el servidor. Esto permite acelerar la carga del sitio web en un 40% y con un 90% menos de JS.
- Cero JS, de forma predeterminada: si no es necesario, Astro no enviará JS al cliente. Solo enviará lo mínimo necesario en caso de que utilicemos la técnica anteriormente mencionada de hidratación parcial.

[2]https://astro.new/

- Desplegable en cualquier lugar: ya sea en entornos de ejecución global Edge, como Deno o Cloudflare, o en cualquier otro.
- Personalizable: podemos integrar frameworks, lenguajes, librerías, etc. Por ejemplo Tailwind, Markdown, TS o React.

III. Desarrollar con Astro

A. Estructura del Proyecto

En esta sección veremos cómo es la estructura de un proyecto en Astro:

- .vscode: configuración del editor vscode y la extensión de Astro.
- node_modules: dependencias y módulos que necesita el proyecto para funcionar.
- public: archivos que queremos que sean accesibles desde el navegador en producción.
- src: código y archivos de desarrollo, no accesibles en producción.
- dist: al ejecutar el comando *npm run build*, se crea esta carpeta con los archivos ya procesados y listos para el entorno de producción.
  Además de otros archivos habituales como .gitignore, astro.config.mjs, package.json, README.md y otros.

B. Generar Múltiples Páginas

Para desarrollar una MPA, debemos crear archivos **.astro** en el directorio **src/pages**. Después, estas páginas se renderizarán a HTML antes de producción. Los archivos .astro son muy similares al formato de los documentos HTML y permiten desarrollar HTML y CSS en el mismo archivo y luego Astro los separa automáticamente en producción. Además, de la misma forma que React, estos archivos permiten el uso de **sintaxis JSX, Markdown e importación de componentes de otros** *frameworks*.

C. Sintaxis JSX

Los archivos .astro permiten escribir código JS exclusivamente para la fase de desarrollo. Si utilizamos la siguiente sintaxis:

```
---
const saludo = "Hola"
---

<h1>{saludo}</h1>
```

Dentro de los guiones podemos escribir código JS al que, posteriormente, podemos acceder con las llaves. Pese a que esta sintaxis de JS acabará convertida en texto HTML, nos puede ser útil para cambiar clases CSS y otras propiedades de forma ”dinámica”, atendiendo a condicionales y eventos.

En el interior de las llaves podemos utilizar el operador ternario, los operadores *and* y *or* y similares para generar condiciones. Sin embargo, no podemos hacer uso de **if, for**, etc.

Otra aplicación interesante de esta sintaxis es la posibilidad de utilizar funciones como **map** y **forEach** para gestionar *arrays* provenientes del *back-end*.

D. Top-level Await

Top-level await es una característica de Astro que nos permite hacer lo siguiente:

```
---
await fetch(<url>)
---
```

Gracias a la característica mencionada, podemos observar que no es necesario incluir ningún **async**. Esto se debe a que se incluye por defecto encima de todo el código JS.

E. Componentes de Astro

De la misma forma que Vue, Astro nos permite crear porciones de interfaz reutilizables o componentes. Además, Astro nos permite declarar *props* para pasarle información a cada componente desde el *back-end* y especificar con código TS el tipo de variables que se reciben (*interface*).

Esto también nos permite crear *Layouts* para evitar repetir el código HTML5 de inicialización o los estilos CSS.

F. Integración de Componentes

Astro permite integrar componentes de otras librerías o *frameworks* como Vue, React, Solid, etc. Esto es posible debido a que Astro está desarrollado sobre Vite JS, una herramienta de *front-end* que nos ayuda a crear proyectos sin atarnos a ningún *framework* concreto.

Sin embargo, es importante recalcar que nuestros componentes solo se renderizarán en el servidor como HTML estático, a no ser que hidratemos el componente a dinamizar. Esto se hace mediante directivas de hidratación como *client:load*, que carga el JS para el componente a hidratar justo al cargar la página.

IV. Conclusiones

En conclusión, Astro es un *framework* de desarrollo web todo en uno que combina la comodidad y la facilidad de uso con un rendimiento óptimo y una carga rápida de las páginas. Su enfoque en el contenido, la arquitectura MPA y la capacidad de integración de componentes de otros *frameworks* lo convierten en una opción atractiva para desarrolladores que deseen crear sitios web estáticos, rápidos y personalizables. Astro ofrece un conjunto de características poderosas y una sintaxis familiar que facilita la creación de proyectos web de alta calidad.

Referencias

[1] J. Pradas (2023, May). Página Web del Proyecto [Online]. Disponible en: https://fir3l1ght.github.io/astro-landingpage/

[2] J. Pradas (2023, May). Repositorio de GitHub del Proyecto [Online]. Disponible en: https://github.com/fir3l1ght/astro-landingpage

[3] Fazt (2023, Ene). Curso de Astro, Generador de Sitios Web Estáticos [Online]. Disponible en: https://youtu.be/sOXW0ZnJxbQ

[4] Astro Team. Documentación de Astro [Online]. Disponible en: https://docs.astro.build/es/getting-started/

[5] midudev (2022, Sep). Hilo de Twitter Acerca de Astro [Online]. Disponible en: https://twitter.com/midudev/status/1567859756303269895?lang=es

[6] LearnVue (2022, Ago). Vue and Astro Simplified [Online]. Disponible en: https://www.youtube.com/watch?v=1BU12Z5w_FU

[7] M. Domínguez (2021, Nov). Vue.js + Astro – ¿Mejor que una SPA de Vue? [Online]. Disponible en: https://todoxampp.com/vue-js-astro-mejor-que-una-spa-de-vue/
[8] Vue Team. Built-in Directives [Online]. Disponible en: https://vuejs.org/api/built-in-directives.html#v-else
[9] unDraw. unDraw [Online]. Disponible en: https://undraw.co/search
[10] You, Evan y otros colaboradores. Vue.js [Online]. Disponible en: https://vuejs.org/
[11] Microsoft. TypeScript [Online]. Disponible en: https://www.typescriptlang.org/
[12] Gruber, John. Markdown: Syntax [Online]. Disponible en: https://daringfireball.net/projects/markdown/
[13] You, Evan y otros colaboradores. Vite [Online]. Disponible en: https://vitejs.dev/
[14] Meta. React [Online]. Disponible en: https://react.dev/